Transactions in GIS, 2005, 9(4): 561–583

Research Article

Adaptable User Profiles for Intelligent Geospatial Queries

Giorgos Mountrakis
Department of Spatial Information Science and Engineering, and National Center for Geographic Information and Analysis University of Maine

Peggy Agouris
Department of Spatial Information Science and Engineering, and National Center for Geographic Information and Analysis University of Maine

Anthony Stefanidis
Department of Spatial Information Science and Engineering, and National Center for Geographic Information and Analysis University of Maine

Abstract
The geospatial information user community is becoming increasingly diverse, with numerous users accessing distributed datasets for various types of applications. Currently in GIS, unlike traditional databases, there is a lack of machine learning algorithms to customize information retrieval results. Thus the particular interests of individual users are not taken into account in traditional geospatial queries. In this paper we present a system that adjusts query results based on user requirements and needs. It does so by using a collection of fuzzy functions that express user preference specifically in GIS environments. The focus of this work is on preference learning for one-dimensional, quantitative attributes, and on the customization of geospatial queries using this information. The model used to express user preferences adjusts gradually to the underlying complexity during a training process, starting with fairly simple linear functions and progressing to complex non-linear ones as needed. Our advanced modeling capabilities are demonstrated through an applicability example, and statistical simulations show the robustness of our system.

Address for correspondence: Giorgos Mountrakis, Department of Environment Resources and Forest Engineering, State University of New York 312 Bray Hall, Forest Drive, Syracuse, NY 13210. E-mail: gmountrakis@esf.edu

1 Introduction

The development of novel sensors and innovative data acquisition methods, and advancements in computer storage and access capabilities have resulted in tremendous increases in the number of geospatial datasets currently available to the corresponding user community. Parallel to these developments, the user community itself is undergoing an expansion and transformation, with constantly increasing user numbers and applications that require access to geospatial information. This emerging reality is affecting geospatial information retrieval (IR) as the same geospatial database may be accessed by diverse users with diverse needs and interests. For example, a surveyor and a meteorologist may have different preference patterns as they aim to retrieve a satellite image depicting an area of interest at a specific instance (e.g. July 2004). If this specific dataset were unavailable, the surveyor would then prefer the most recent image (e.g. June 2004) to accomplish his/her task of updating the 2002 cadastral records. The meteorologist on the other hand is interested in monitoring a specific meteorological event of short duration (e.g. a hurricane), and in that case month old imagery would be useless. In a similar manner the preferences of these two users may differ in terms of other attributes (e.g. resolution, coverage). Currently available information retrieval approaches in GIS typically do not allow the incorporation of user preference in the query process. In order to overcome this shortcoming we have developed an approach that adjusts the ranking of query results based on similarity *profiles* that express user preferences. By doing so users receive results that better match their specific needs and preferences, thus improving the information retrieval process and saving valuable user time. These profiles may be user dependent, application dependent, or both. The content of a user profile is a set of parameters (variables) describing user preferences in a mathematical manner.

The idea of employing user profiles in queries is not new to the database community, and machine learning algorithms have been used to support this task. Typically, users go through a training process to make the computer understand their demands. Such training processes may include user preference modeling within each attribute, determination of the significance of each attribute to the overall solution, and possible correlations (relationships) between attributes. After training, the computer "memorizes" this preference and stores it as a *profile*. Each time a user queries the database an appropriate profile is used to customize query results. For example, in image processing applications, Mitaim and Kosko (1998) proposed a neuro-fuzzy approach with agent profiles using *if-then* rules to optimize texture matching. However, user profiles have not been used in geospatial queries, partially due to the high complexity and variability of preferences exhibited by GIS users. On the other hand, geospatial attributes (e.g. scale, time, coverage) are quantifiable and continuous parameters, and thus are highly suitable for user profile modeling.

In this paper we learn user preferences on quantitative attributes that accompany geospatial information, and focus in particular on one-dimensional attributes (e.g. time, resolution, azimuth). Whereas research in information retrieval has focused on aggregating attribute preferences, we focus instead on modeling user preference within each attribute. This allows us to better model the complexities and diversities of user preferences in geospatial applications. Aggregating single-dimensional information into higher dimensional attributes is beyond the scope of this paper. Nevertheless, the results of our modeling may (and should) be used as input in aggregation algorithms to enable comprehensive modeling.

To accomplish our goal we make use of fuzzy membership functions, as they were first presented by Mountrakis and Agouris (2003). Here we focus on the applicability of the method for profile creation and provide a detailed statistical evaluation of the process. We also show how we support various levels of user expertise in both the training and the simulation processes. The remainder of the paper is structured as follows. In the next section we provide an overview of existing methodologies. In section 3 we position our system in the overall process for information retrieval from geospatial attributes. The corresponding similarity functions used are also discussed and the applicability within the geospatial domain is justified. Section 4 analyzes how the user interacts with our system, in other words the input and output of the process. Section 5 discusses profile creation and use, and statistical tests are presented in Section 6 to verify the robustness of our method. Section 7 provides the conclusions of this work.

2 Related Work

The idea of training a computer to understand human concepts to later incorporate them in analytical tasks is a major goal of machine learning within the field of artificial intelligence (AI). Relevant applications of AI concepts in GIS include efforts to identify relationships between GIS objects (Walker and Moore 1988), to perform feature extraction from digital elevation models (DEM) (Bennet and Armstrong 1996), to support texture matching in aerial photographs (Ma and Manjunath 1998) and to perform road extraction from multi-spectral imagery (Doucette et al. 2001).

Similarity assessment in database queries typically involves the representation of stored information as points in a multidimensional attribute space, and the use of a distance metric to estimate similarity between these points and a query request (Aha 1992). There is no single distance function to offer optimal results in all applications. This led to a variety of functions over the years, some more general and others more application specific (Table 1). In Table 1 $\vec{x}$, $\vec{y}$ are the input vectors whose distance is assessed (e.g. one being a stored record and the other a query input), and m is the number of examined variables. Matrices Q and V are of size $m \times m$ and they represent a problem-specific positive definite weight matrix and a covariance matrix, respectively. Variables $\bar{x}_i$, $\bar{y}_i$ are the average values for attribute i occurring in the training set. Variable sum_i is the sum of all values for attribute i occurring in the training set and $size_x$ is the sum of all values in vector $\vec{x}$.

Distances are often normalized by dividing the actual distance metric by the corresponding range (max – min values). In addition to normalization, attribute weights (e.g. quadratic distance metric) and other weighting schemes have been incorporated in the learning process (e.g. Wettschereck et al. 1995, Atkeson et al. 1997). Additional distance functions include the context-similarity measure (Biberman 1994), the contrast model (Tversky 1977), and the hyper-rectangle distance functions (Salzberg 1991, Domingos 1995). Popular similarity measures for documents are the cosine measure, the Pearson correlation, and the Jaccard similarity function (Leydesdorff 2004). For nominal attributes the Value Difference Metric (VDM) was introduced by Stanfill and Waltz (1986) and variants followed (Cost and Salzberg 1993, Rachlin et al. 1994, Domingos 1995). In Wilson and Martinez (1997) three more functions were introduced, namely the Heterogeneous VDM, the Interpolated VDM, and the Windowed VDM.

Table 1 Popular distance functions and their equations

Name	Equation
Minkowsky (Batchelor 1978)	$D(x, y) = \left(\sum_{i=1}^{m} \lvert x_i - y_i \rvert^r \right)^{1/r}$
Euclidean (Created for $r = 2$ at Minkowsky function)	$D(x, y) = \sqrt{\left(\sum_{i=1}^{m} \lvert x_i - y_i \rvert^2 \right)}$
Manhattan or *city-block* (Created for $r = 1$ at Minkowsky function)	$D(x, y) = \left(\sum_{i=1}^{m} \lvert x_i - y_i \rvert \right)$
Quadratic (Adding a weight matrix to Euclidean)	$D(x, y) = (\vec{x} - \vec{y})^T Q (\vec{x} - \vec{y})$
Mahalanobis (Nadler and Smith 1993)	$D(x, y) = (\det V)^{1/m} (\vec{x} - \vec{y})^T V^{-1} (\vec{x} - \vec{y})$
Camberra (Lance and Williams 1966)	$D(x, y) = \left(\sum_{i=1}^{m} \left\lvert \frac{x_i - y_i}{x_i + y_i} \right\rvert \right)$
Chebychev	$D(x, y) = \max_{i=1}^{m} \lvert x_i - y_i \rvert$
Correlation	$D(x, y) = \frac{\sum_{i=1}^{m} (x_i - \bar{x}_i)(y_i - \bar{y}_i)}{\sqrt{\sum_{i=1}^{m} (x_i - \bar{x}_i)^2 \sum_{i=1}^{m} (y_i - \bar{y}_i)^2}}$
Chi-square (Diday 1974)	$D(x, y) = \sum_{i=1}^{m} \left(\frac{1}{sum_i} \left(\frac{x_i}{size_x} - \frac{y_i}{size_y} \right)^2 \right)$

The above functions provide a simple model that allows efficient indexing by dimensionality reduction techniques (Gionis et al. 1999, Yi and Faloutsos 2000). However, the simplicity of these functions does not support advanced learning capabilities and high adaptability to complex queries. To compensate for this shortcoming more complex algorithms have been proposed. Some of them involve neural networks with applications in content (Lim et al. 2001), information (Mandl 2000) and image-based (Carkacioglu and Fatos 2002) retrieval. Fuzzy sets have also been used in visual retrieval systems (Santini and Jain 1999) and fuzzy integrals approximate similarity in genetic algorithms (Ishii and Wang 1998).

These methods focus on multi-dimensional similarity aggregation between attributes to provide an overall similarity metric. Our work complements these efforts by addressing similarity modeling within a single attribute, to better represent complex patterns of user preference. In this paper we propose a novel way to:

1. Perform intelligent queries using profiles that express similarity preference within dimensions.
2. Design these profiles to fit specific preference as expressed in geospatial queries.

3 Learning Geospatial User Profiles

In this section we discuss our approach for learning user profiles. In section 3.1 we position our work within the current state of knowledge, while in section 3.2 we introduce the specifics of our method. The last section (3.3) addresses the applicability of our method in the geospatial domain.

3.1 Similarity matching process

In the context of this work, a geospatial information object O is an autonomous entity with a specific database record. Examples may include a map, a DEM, a satellite image, or the record of a building in a cadastre database. Within a database, such objects are typically described by a set of attributes. For example, a satellite image may be described through its coverage, resolution, time and type of sensor, among others. Some of these attributes may be conceptually related and thus may form distinct conceptual groups. Such an example would be grouping separately the metric and the qualitative attributes. This hierarchical arrangement is further explained in Figure 1.

The comparison of an information object O stored in a database to a query request O^q entails the comparison of their corresponding attribute values. This may be a straight forward issue if the attributes used to describe both O and O^q are the same, or may require advanced translation methods (e.g. using ontologies) to establish correspondences among two heterogeneous sets of attributes. Assuming similar representations, the comparison of a stored object to a query request involves the use of matching functions to produce a similarity metric S as:

$$S = g[h_1(t_{11}(f_{11}, f^q_{11}), t_{12}(f_{12}, f^q_{12}), \ldots), \ldots, h_i(t_{i1}(f_{i1}, f^q_{i1}), \ldots, t_{ik}(f_{ik}, f^q_{ik}), \ldots, t_{in}(f_{in}, f^q_{in})), \ldots] \quad (1)$$

where function t_{ik} expresses similarity between a *database attribute value* f_{ik} and the corresponding *query value* request f^q_{ik}; function h_i integrates similarity results from each separate attribute to provide a similarity metric for each conceptual attribute grouping; and function g is the overall similarity measure, combining similarity from each conceptual group into a single total metric. The challenge of intelligent database queries is to define functions t_{ik}, h_i and g so they express user perceptions of similarity.

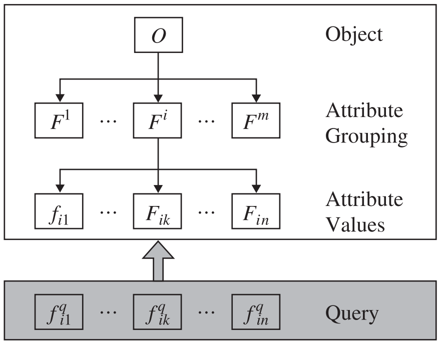


Figure 1 Hierarchical object attribute representation

For example let us consider a query of a geospatial database. Assume that global similarity is calculated based on three attribute groups, namely F = [Metric attributes, Qualitative attributes, Dataset accessibility]. The "Metric attributes" group may be represented by two attributes, namely *time* and *scale* (F^1 = [Time, Scale]). An example of a query request may be $F^{1q} = (f^q_{11}, f^q_{12})$ = [10 am, 50 cm], aiming for the recovery of datasets depicting an area at 10 am, with a scale (resolution) of 50 cm. After this query is presented to the system, similarity within each attribute is assessed using functions t_{ik} that explain how similar the stored values are to the query request. For example, a function t_{11} would calculate the similarity in time between the query request and a database candidate, while another function t_{12} would calculate the similarity in the scale attribute. Subsequently, function h_i would aggregate these similarity results from different attributes to provide a similarity metric for the corresponding attribute group (e.g. producing an aggregate similarity value for the "Metric attributes" group from the similarity results obtained from attributes Time and Scale). In the last step, function g integrates similarity values from different attribute groups to derive an overall similarity metric expressing similarity between the query request and a database record. In many cases functions h_i and g are treated as one function depending on the organization of the attributes.

Currently, the focus of database query research has been on the development of complex non-linear models for functions h_i and g. In contrast, the t_{ik} functions have received little attention and are typically modeled in a relatively simple manner. However, it is easily understood that if the t_{ik} functions fail to describe adequately the corresponding similarity relationships, the resulting integrated similarity metrics would be significantly compromised. In GIS queries user preferences may be much more complex than general queries (e.g. text queries), while the diversity of users and applications further increase the need for efficient modeling of the t_{ik} functions. Thus, modeling user similarity preference within each attribute can substantially help geospatial queries. Motivated by these observations, the focus of our work is to investigate the application of complex functions for similarity preference within each attribute (t functions of equation 1). An applicability example demonstrating the necessity of more complex (e.g. asymmetrical, non-linear) modeling is presented in section 5.1.

3.2 Algorithm Design

The approach we follow for similarity learning within one-dimensional quantitative attributes is based on the learning system introduced in Mountrakis and Agouris (2003). User preferences are modeled as preference surfaces. Such surfaces are expressed in the form of [X,Y,Z], where X is the query value, Y is the database value and Z shows the degree of similarity between X and Y. It is easily understood that the Z value reaches its maximum when the database value is the same as the query value.

Similarity preference can be asymmetrical therefore two separate surfaces are constructed. For example, a database return of an aerial photograph four months before or after the requested time might not have the same appeal to a user. For visualization purposes a preference surface is presented later in Figure 6. To generate such surfaces we follow the method of Figure 2 with the processing steps (shadowed rectangles) explained in the discussion below.

- **User Feedback.** Users are providing input for the training process. More specifically, they are presented with pairs of sample values within each dimension, and are asked

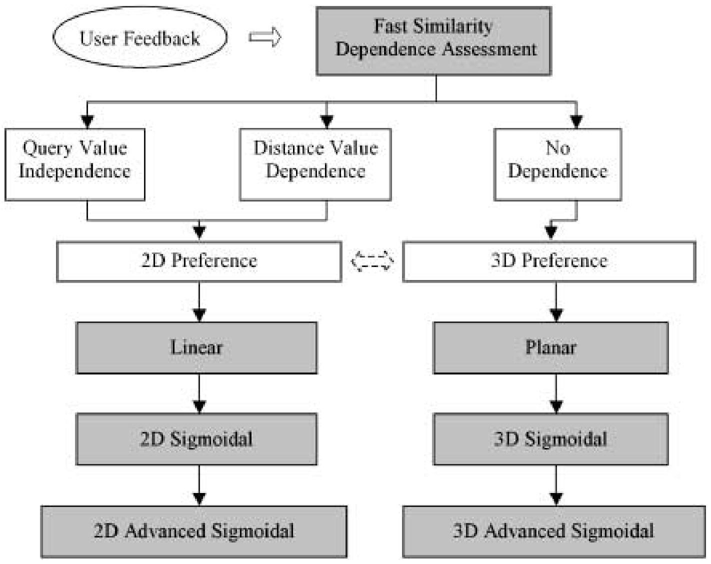


Figure 2 Similarity learning algorithm – Training flow

to assess their similarity. For example, they are asked: "*Your request is for a geospatial object with Time = 11/12/2003. How similar to it would be an object from Time = 02/04/2001?*" In this manner a training set is created, comprising a set of [Query_attribute, Database_attribute, Similarity_value] points (2 inputs, 1 output). More information on user similarity assessment is provided in section 4. We should mention here that our training is progressive, meaning that a small number of initial training points is adequate to initiate our modeling, and based on the underlying complexity more samples are requested when necessary.

- **Similarity Dependence Assessment.** In this initial processing step we attempt to evaluate the underlying complexity that may exist between the abovementioned inputs (Query_attribute, Database_attribute) and output (Similarity_value). More specifically we try to identify whether the given similarity preference is:

1. Independent of the Query Value; or
2. Dependent solely on the distance between Query and Database Value and not the actual Query and Database Values.

The first case occurs when users may not have a clear idea of what their target (i.e. query) value is, so close enough queries will have identical similarity preference. The second, and much more frequent, case identifies user preference that is solely distance-dependent, in other words no matter what the Query Value is, a specific distance between Query and Database Value will always yield the same similarity result. Recognition of either of the two aforementioned dependence cases is important because the dimensionality changes from 3D to 2D, facilitating faster training and simulation times. If no dependence is detected the algorithm proceeds with the more computationally expensive (but necessary) three-dimensional solution with two inputs (Query and Database values) and one output (Similarity value).

A critical aspect of the algorithm design is that it can identify or dismiss dependencies and hence switch from a two to three-dimensional solution and vice versa in real-time (hence the dotted arrow in Figure 2). The fuzzy function equations are formulated in such a way that this switch can occur naturally through a single parameter change in the corresponding functions. In the next sections we describe the three-dimensional solution, since the two-dimensional is a simplified derivative of it.

- **Planar Preference Surface.** This is the simplest set of fuzzy membership functions (FMFs) used, and comprises two independent, piecewise planar composite surfaces to support asymmetrical cases. Each of these two composite surfaces is formed by a series of planes. We use five planes to model each surface, with each plane expressing similarity within a certain range. Plane information provides approximate parameter values for the next more complex functions, like the sigmoidal one that follows.
- **Sigmoidal Preference Surface.** When user preference patterns are too complex to be represented by a planar surface, we proceed with a more complex modeling using a sigmoidal function. Sigmoidal functions are popular in the neural network community and have been used in the GIS field as predefined similarity functions for spatiotemporal trajectory matching (Vlachos et al. 2002). Our similarity surface comprises two distinct sigmoidal functions to compensate for asymmetrical cases.

An important characteristic of the sigmoidal function is the large range of modeling capabilities. Efficient manipulation of the slope parameter can result in representing a variety of cases, ranging from a linear to a step-like behavior. This diverse capability together with the large operational range on the input space and the mathematical continuity of the function (first derivative exists everywhere) establishes the sigmoidal as an appropriate preference surface from a variety of available fuzzy membership functions.

- **Advanced Sigmoidal Preference Surface.** When the underlying complexity of the problem is even higher, the already presented similarity functions may not be able to model it adequately. For these cases we present a more adaptable set of functions with higher modeling capabilities. We do so through functions of higher input dependency. Such a case might exist when users are more tolerant (in a non-linear fashion) towards query deviations as the query value increases. An example is presented in section 5.1.

3.3 Applicability within the Geospatial Domain

Our approach to user preference modeling is influenced by the unique characteristics of geospatial information that differentiate it from other types of information. Of primary interest to this work is the fact that one-dimensional geospatial parameters are quantifiable and continuous. Thus a user may easily express his/her preference as functions of such quantifiable and continuous parameters. Considering the example of resolution, a user may state and quantify preferences through the following statement: *I am interested in imagery with a resolution of 50 cm, and my interest drops linearly/exponentially as resolution decreases*. In this manner, a GIS user can quantify expressions of preference in terms of resolution, making for example an aerial photograph with 2 m resolution twice as suitable for his/her application than another with 4 m resolution. Thus preference expressions in one-dimensional geospatial parameters are not only *ordinal*, but *metric* as well.

This richness and metric structure of geospatial information (and corresponding user preference patterns) is a very important aspect that differentiates geospatial from other types of information collections (e.g. text databases, stock prices). Of course, there still exist a number of properties that may be included in a GIS but do not have such structure (e.g. land use, ownership), but these attributes are beyond the scope of this paper. Our focus is on one-dimensional quantitative attributes (e.g. scale, resolution, area, azimuth) and corresponding complex user preferences.

4 Expressing Similarity in Various Resolutions

An important characteristic of our approach is its ability to support a variety of classification schemes that express similarity. This can be accomplished at the input level during training, compensating for diverse user expertise, and at the output level during simulation. Naturally, the algorithm's accuracy is improving as the similarity resolution for the classification becomes more detailed.

In order to train our system we present users with a set of (Query_attribute, Database_attribute) values and request a similarity assessment. This assessment is performed in the form of a ranked classification by choosing a class describing user perceptions of similarity. User preference is expressed through a crisp set of n classes [class_1, class_2, . . . , class_n] that represent similarity within the [0,1] interval. These classes may be linguistic terms, also known as linguistic hedges (Zadeh 1972) or may be expressed by a numerical descriptor. In order to facilitate users with various similarity assessment capabilities (novel-to-expert) numerous classification schemes can be used. The selected classification scheme does not affect our algorithm's applicability, but it is easily understood that finer classification schemes would result in higher accuracies. For example, a linguistic classification scheme for a novel user could be:

Similarity Classes = {No Similarity, Very Low, Low, Average, High, Very High, Identical}

A more expert user could use a higher number of classes (e.g. 13), following for example a university grading scheme like this one:

Similarity Classes = {F, E, D^-, D, D^+, C^-, C, C^+, B^-, B, B^+, A^-, A}.

For the purposes of our application we assume a *linear ordering* of these classes, since we are modeling similarity preference of each specific user (we do not combine preference from different users). Partial ordering is not examined because all similarity values correspond to a single attribute evaluation (we do not combine attributes), therefore all similarity values are directly comparable to each other. Of course there exists incomparable information in the real world within which there are linguistic terms that do not correspond to a linear ordering on the universe (Xu et al. 1999) but such research is outside the scope of our work, as we assume the user is constrained to select among a predefined set of classes.

Our algorithm takes the ranking provided through the set of n classes and translates it into a quantitative similarity value. For linearly ordered classes, a similarity value is assigned to each class using the following equation:

$$V_i = 100 * (i - 1)/(n - 1) \tag{2}$$

where n represents the total number of classes, i the current class under examination and V_i the quantitative similarity representation for class i. The algorithm performs the necessary training based on these quantitative values.

4.1 Class Identification after Simulation

After a successful preference surface is found the model is used for simulation, during which the system calculates the corresponding similarity value for a new case. We compare this calculated similarity value with the pre-defined quantitative similarity representations of each class (from equation 2). In most cases an exact match will not exist between these values to assign automatically a single class as a return. Our system can then follow one of these two approaches:

- Use a minimum distance criterion to match the calculated similarity value to a single class; or
- Return not a single class, but both classes that the value is closest to.

An example can be seen in Figure 3. Let us assume that the number of similarity classes is five, namely *Classes* = {*No Similarity, Low, Average, High, Identical*} and their corresponding quantitative transformation using equation 8 is *Classes* = {*0%, 25%, 50%, 75%, 100%*}. Now let us examine the case of a calculated similarity value of 42% (or 0.42) for an input set, which is a value that does not match any of the five quantitative values representing the classes. The system could return the class that is closest in the quantitative value, which in this case would be the "*Average*" class. Alternatively, the system would return not a single class, but both classes that the quantitative value falls between, in this example classes "*Low Similarity*" and "*Average Similarity*".

Here we should point out that, strictly speaking, a direct class identification would be incorrect. The minimum distance criterion used to identify the winning class is based on the fact that a metric distance has been established between classes during training. For the above example, however, this is not the case: the user has provided only a ranking (ordering) between the classes, not an exact relationship between them. We know that

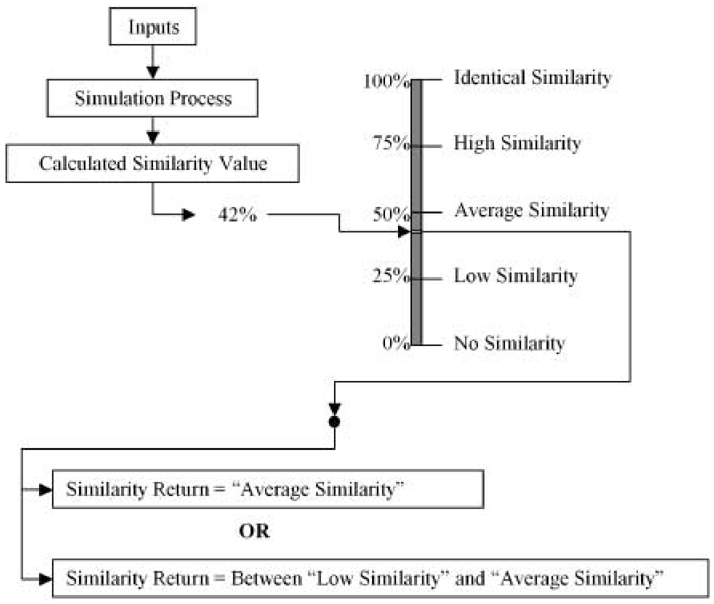

Inputs
Simulation Process
Calculated Similarity Value
42%
100%
75%
50%
25%
0%
Identical Similarity
High Similarity
Average Similarity
Low Similarity
No Similarity
Similarity Return = "Average Similarity"
OR
Similarity Return = Between "Low Similarity" and "Average Similarity"

Figure 3 System return values

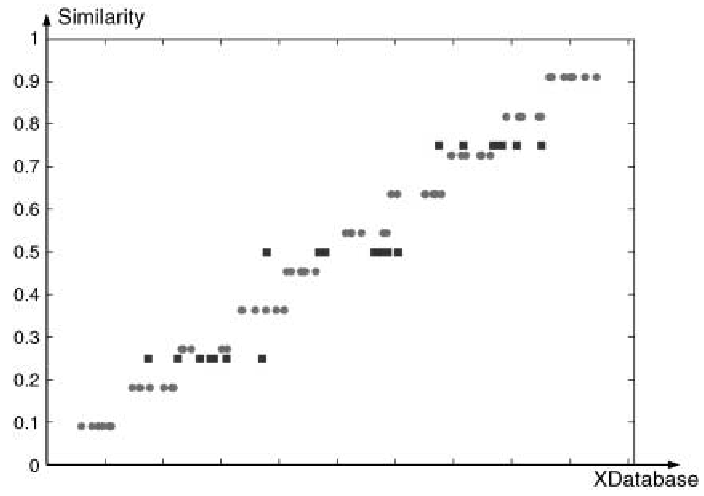


Figure 4 Effect of number of classes on training set

class "*Low Similarity*" expresses a smaller similarity degree that the "*Average Similarity*" class but we do not know that "*Low Similarity*" is twice as worse than "*Average Similarity*" as the quantitative values might be implying. So the distance criterion might be misleading in some cases and should be used with caution.

4.2 Influence of Number of Classes

The approach described above requests a classification from the user at the training stage and returns a classification at the simulation stage. The number of classes chosen does not affect the applicability of the algorithm. Several classification schemes can be used based on user expertise and modeling accuracy demands. The higher the number of classes the more detailed the input is and the more advanced the modeling capabilities that can be reached. It is easily understood that a very limited number of classes would lead to generalization in parts of the output space (similarity) where no information exists. This can be easily understood by examining Figure 4. This graph represents a set of inputs with different numbers of classes (0% and 100% similarity values are omitted). With dark gray squares we can see the training set that would be provided by three classes, while light gray circles represent the corresponding set for10 classes. It is evident that the higher the number of classes, the better the output resolution (similarity) is, and therefore generalization errors are restricted.

5 Creation and Use of Geospatial Profiles

In this section we present an example to show how each profile is created and to illustrate the advanced modeling capabilities exhibited by our similarity preference learning algorithm. We also present the content of each profile and discuss their reusability.

5.1 Applicability Demonstration

In this section we demonstrate the use of our method using the *scale* (or equivalently, *resolution*) dimension. The example is expanded from Mountrakis and Agouris (2003), to facilitate the in-depth understanding of our method by adding a step-by-step explanation and representing the corresponding outputs of each process.

Let us consider user requests for imagery of a particular resolution (pixel ground size). In general, user preferences may be inherently asymmetrical and non-linear with regard to the resolution attribute. Assume for example that the application motivating this request is road mapping. With decreasing image resolution user preference decreases gradually and not necessarily in a linear fashion. When resolution becomes coarser than a certain limit, a smooth road segment is represented as a succession of step edges and is therefore unsuitable for mapping. This reduces user preference at a nearly exponential rate once a certain resolution threshold is reached.

On the other hand, finer resolution may not necessarily imply a user preference of 100%. Cost considerations such as price, storage, and processing time may cause user preference to decrease (linearly or non-linearly) as resolution increases. In general, these two patterns of user preference variation as resolution deviates from the ideal value may be asymmetrical.

Figure 5 shows the process followed to capture the above similarity preference with our fuzzy membership functions:

- Create a training dataset by providing the user with different pairs (query and return) of pixel size and request a similarity metric;
- Interpolate planes on each half ($X_{\text{Query}} > X_{\text{DB}}$ and $X_{\text{Query}} \leq X_{\text{DB}}$) and calculate the corresponding plane parameters;
- Examine whether the interpolation error is better than the desired accuracy;
- If not, calculate angle φ and its associated error;
- If angle φ is close (closeness range predefined by user) to 45 or 90 degrees then eliminate the third dimension (for more information see Mountrakis and Agouris 2003);
- Interpolate two sigmoidal functions each applied on the corresponding half and calculate initial approximations of the sigmoidals using the plane parameters;
- Re-examine whether the interpolation error is better than the desired accuracy;
- If not, allow the so far constant sigmoidal parameters to be expressed by more complicated functions as pre-selected by the user;
- Solve for the parameters including the additional variables; and
- Return the best possible preference surface from the above functions.

In Figure 5 the contour plots of the resulting similarity surface from each fuzzy function are presented. After training for our scale example, the resulting function in the right half is a sigmoidal one with a variable slope *a* (Figure 5 bottom contour graph). The variable slope is able to express user-alterable tolerance depending on their asking pixel size. The larger the requested pixel size the more flexible they are about additional pixel size. That does not happen in a linear fashion so a gradual decrease of *a* (slope) can model that. In the left half the rate of similarity decrease does not change so slope *a* remains the same (i.e. the isoline distance). What changes is the spread value *c* so this complex fuzzy function has the ability to express user similarity tolerance as query value increases. The final similarity surface is presented in Figure 6.

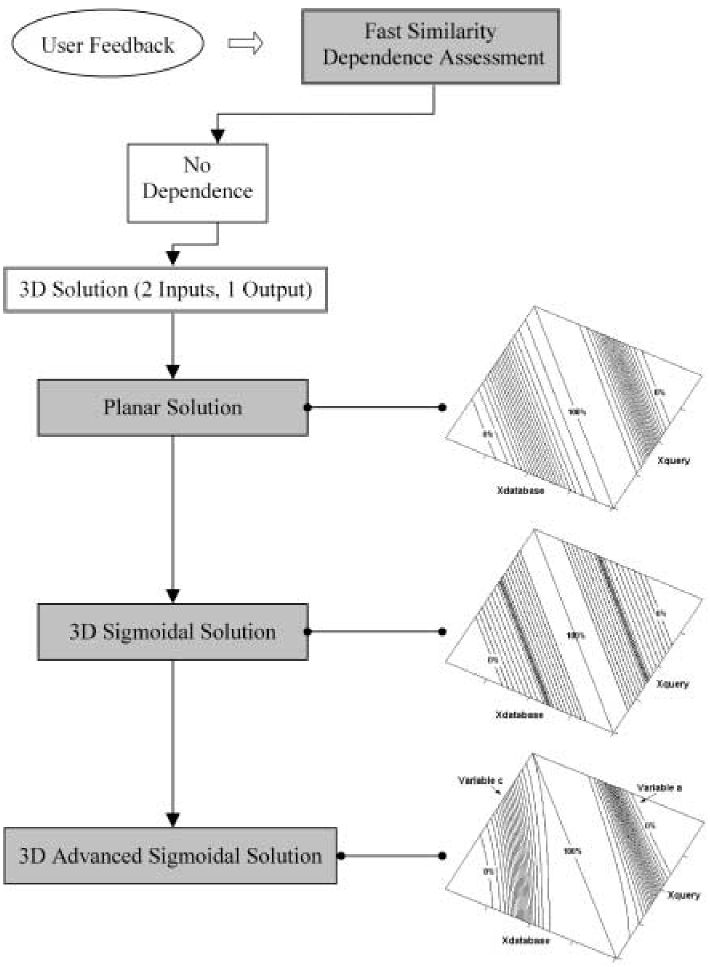


Figure 5 Scale attribute profile training

5.2 Content and Management of User Profiles

Using the above described process we generate user profiles describing user-specific similarity preferences. Such profiles may be stored as visualized in Figure 7. *Profile ID* is a unique key number to identify the specific profile (typically corresponding to a specific combination of user and application). For each of the two asymmetrical solutions a separate function is provided. The *Class ID* refers to the type of function used and the relationship between the parameters and the function's output (i.e. the similarity equation). The calculated *parameters* of each function are stored thereafter. By collecting numerous profiles we can generate a profile library. Even though the management of such a library is beyond the scope of this paper, it is appropriate to point out some important emerging opportunities. Firstly, user profiles may be labeled with *user_background* and *application* information. For example, a user may identify him or herself as '*Biologist*' and select '*watershed identification*' as a specific application, and then proceed to generate the

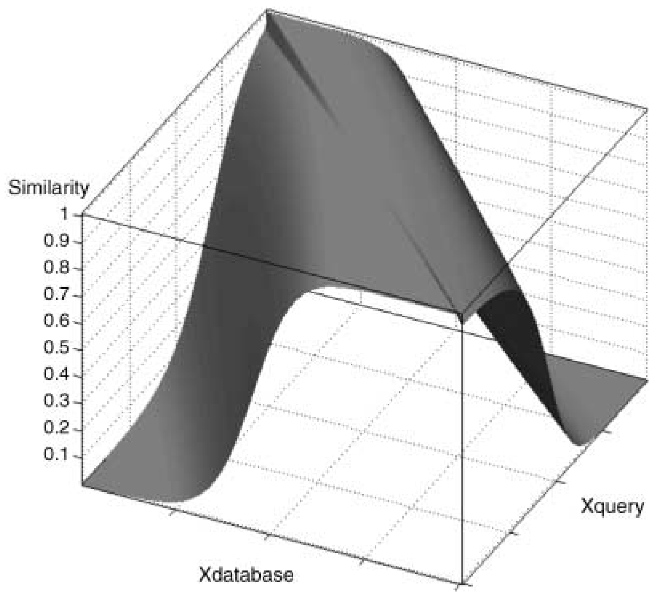
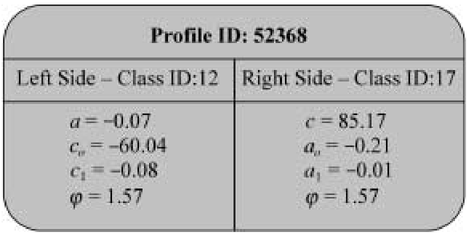


Figure 6 Advanced sigmoidal fuzzy similarity function



Profile ID: 52368	
Left Side – Class ID:12	Right Side – Class ID:17
$a = -0.07$ $c_o = -60.04$ $c_1 = -0.08$ $\varphi = 1.57$	$c = 85.17$ $a_o = -0.21$ $a_1 = -0.01$ $\varphi = 1.57$

Figure 7 Profile example

corresponding profile. Subsequent users that share identity and/or application may access this profile and adopt it for their queries, thus bypassing the profile generating process. A second opportunity is related to the fact that various profiles that display adequate similarity may be grouped together to identify user clusters that display similarity in terms of data needs and query preferences. Through such a grouping we may be able to identify similarities in needs across dissimilar communities and use this information to share or re-organize dataset collections.

6 Statistical Evaluation

An important question that often rises when dealing with complex non-linear function approximations is the stability of the algorithm. In other words we investigate the influence of factors such as initial approximations, noise and number of training samples on the least squares solution. A thorough investigation is presented below to assess algorithm performance.

6.1 Influence of Initial Approximations

A repetitive problem in non-linear least squares solutions is caused by the fact that if the initial approximations are far from the target values there is the possibility that convergence to the desired preference surface will not always be achieved. To address this within our system, we have developed a method of calculating accurate initial approximations from previously interpolated less complex functions. Here we investigate the relationship between convergence and the distance between initial and target values for each of the three parameters: shift, angle, and slope. All tests were performed with 10 training points, and the initial values of the other two parameters having the same value as their corresponding target values so they would not influence the solution. The solution proceeded by estimating all three parameters simultaneously each time to ensure overall stability. Convergence is achieved when solution parameters are within these (strict) thresholds: ±1% of target value for shift, ±0.3 degrees for angle, and ±0.005 for slope.

- **Shift parameter.** The influence of shift approximations can be seen in Figure 8. On the left hand side of this figure we have a graph where the X axis represents the initial approximation value, and the Y axis shows the actual value. The diagonal connecting the upper left and lower right corners of this graph would reflect perfect approximations (equal to the true values). As we move away from this diagonal the quality of approximations deteriorates (they deviate more from the true value). On the right hand side of Figure 8 we have a column displaying the color-coding scheme used for the visualization of convergence success: light gray indicates convergence rates that approach 100%, while dark gray indicates lower convergence rates (eventually

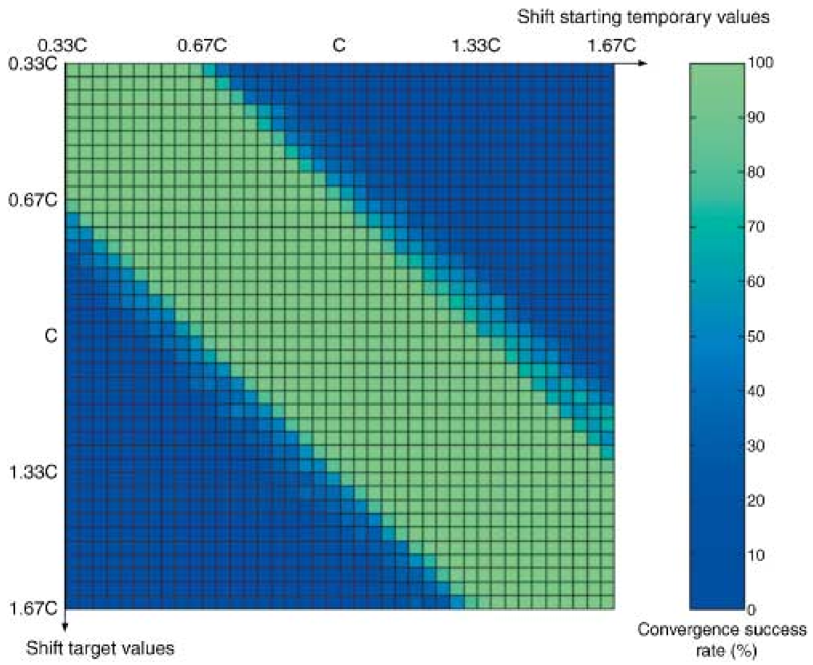


Figure 8 Influence of shift's initial approximations to convergence
This figure appears in colour in the electronic version of this article and in the plate section at the back of the printed journal

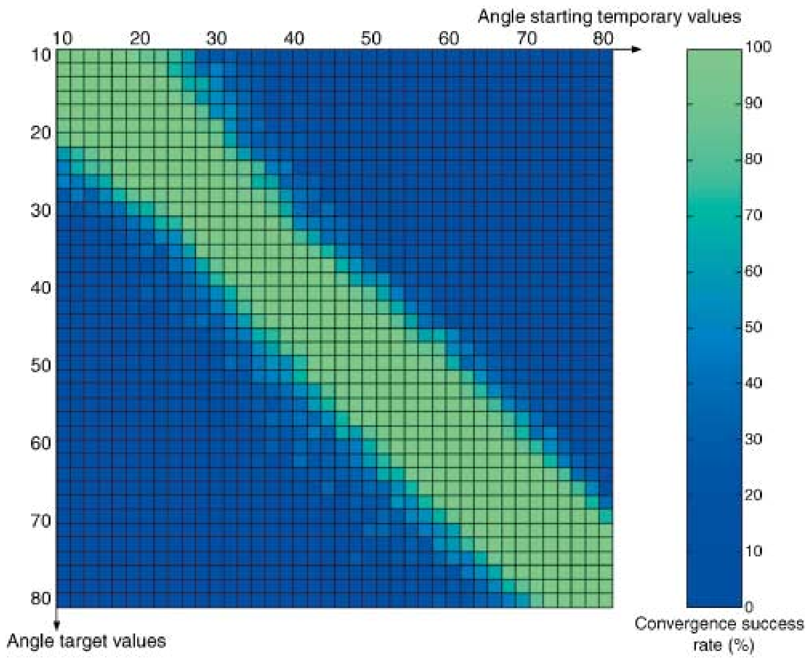


Figure 9 Influence of angle's initial approximations to convergence
This figure appears in colour in the electronic version of this article and in the plate section at the back of the printed journal

approaching 0%). Using this color-coding we can observe in the left hand graph of Figure 8 that approximations within the range of ±33% of the true value C (a random shift value) result into consistent convergence (success rate above 90%). Beyond this threshold we have a rapid drop, with convergence success below 30%. This demonstrates the importance of good approximations in the shift parameter.

The same color-coding scheme and presentation format is used in the subsequent figures to model the effects of other approximations on convergence rate.

- **Angle parameter.** Another parameter we evaluated was the angle, with results visualized in Figure 9. Again, the X axis represents values of initial approximations for the angle parameter, while the Y axis corresponds to the corresponding target value (angles in degrees). We can observe similar behavior to the shift parameter: excellent convergence for approximations that are as accurate as ± 8 degrees, with a rapid drop in our convergence rate for poorer approximations.
- **Slope a.** The most intriguing parameter in our assessment was the slope of the sigmoidal function. The obtained results are presented in Figure 10. The X axis represents values of initial approximations of the slope parameter, while the Y axis represents the corresponding target values. For this parameter we can observe a different behavior than the ones visualized in Figures 8 and 9. More specifically, for small slope target values (<0.15) there is a gradually increasing range of convergence. Beyond the 0.15 target value mark convergence is achieved consistently almost independently of the initial approximation. This is an exceptional result for our algorithm's design because the slope parameter is the only parameter for which we cannot estimate

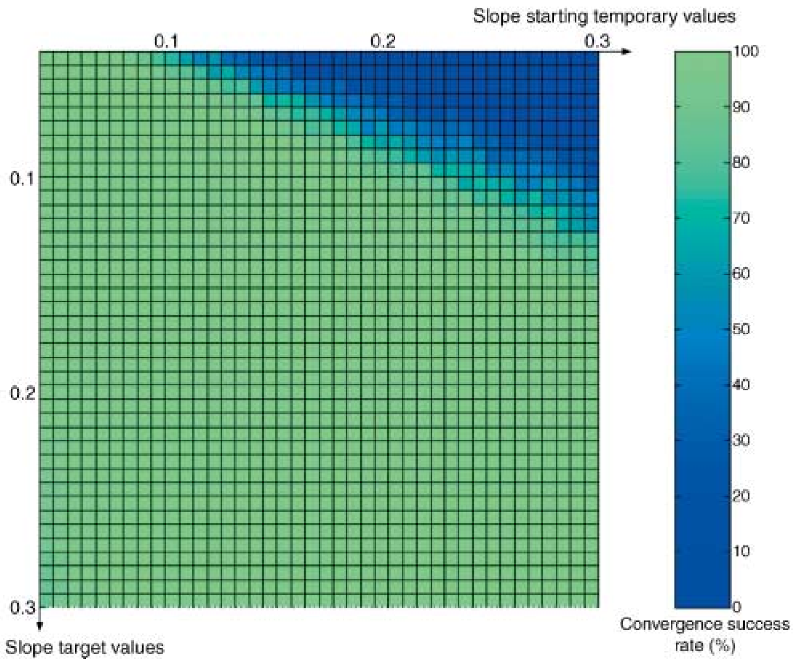


Figure 10 Influence of slope's initial approximations to convergence
This figure appears in colour in the electronic version of this article and in the plate section at the back of the printed journal

good initial approximations. The exceptional tolerance displayed by our algorithm to the quality of slope approximations is highly desirable, as it allows us to overcome this shortcoming. Based on our findings in Figure 10 we assign slope starting values close to zero to achieve high convergence rates independently of the target values.

The explanation for slope's unusual behavior is found in Figure 11, where sigmoidals with different slope values are presented. We can see that a 0.01 change in the slope

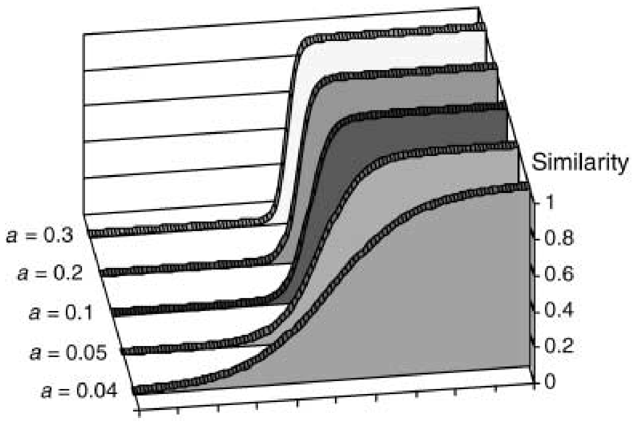


Figure 11 Influence of slope to sigmoidal's shape

value will have a much more drastic influence as the slope gets closer to 0. Therefore changes beyond the 0.15 mark are almost insignificant, which explains the unusually high convergence rate and its independence from the initial value.

6.2 *Influence of Noise and Training Sample Size on Parameter Estimation*

The experiments of this section examine the influence of noise combined with the number of training points leading to a successful solution. Noise was inserted in the training dataset by altering the similarity value of a single point. This alteration varied from 0 to ±0.5 and was imposed on a randomly selected point. The absolute value of similarity change is represented on the X axis of Figures 12, 13, and 14. The Y axis shows the percentage difference from the desired target value in each parameter before noise was added. We should mention that all iterations started with a value 20% away from the target. Experiments were performed for a variety of training sizes, $n = [10, 20, 30, 40, 50]$, to assess the stability of the algorithm. A total of 1,000 iterations were performed for each result and their average is presented in the graphs.

The overall impression from the three figures is that the higher the number of training samples the more tolerant the solution is to noise. This was expected as an outcome for our statistical simulations. Furthermore, the slope parameter seems to be the one most affected with the introduction of noise. This does not necessarily translate to unsuccessful modeling since as we explained in Figure 11 the influence of slope differences is in some parts significant and in others negligible. For this specific test slope was set to 0.07. The other two parameters, shift and angle, appear to be more tolerant to noise with the shift showing a slightly better performance. There is a significant gain when the training size is increased from 10 to 20 points. We should mention of course that we want to keep this number as low as possible to avoid overwhelming the user with an excessive training set.

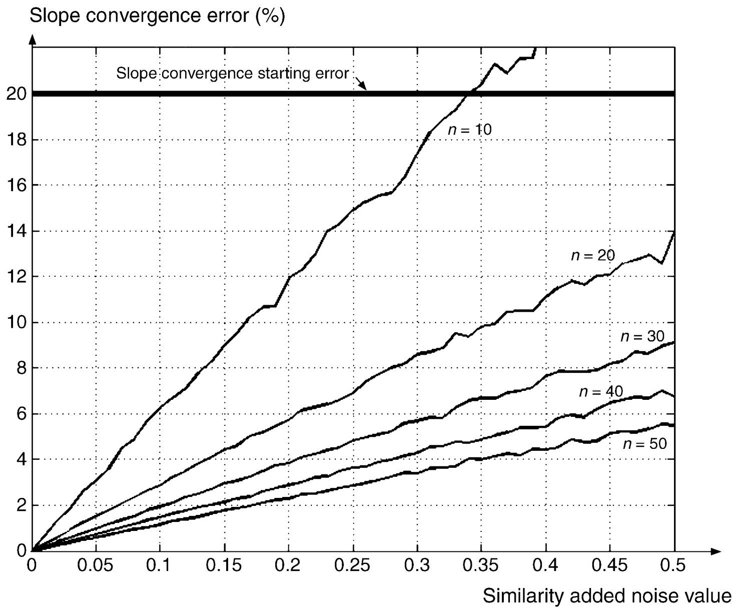


Figure 12 The influence of noise in slope calculation

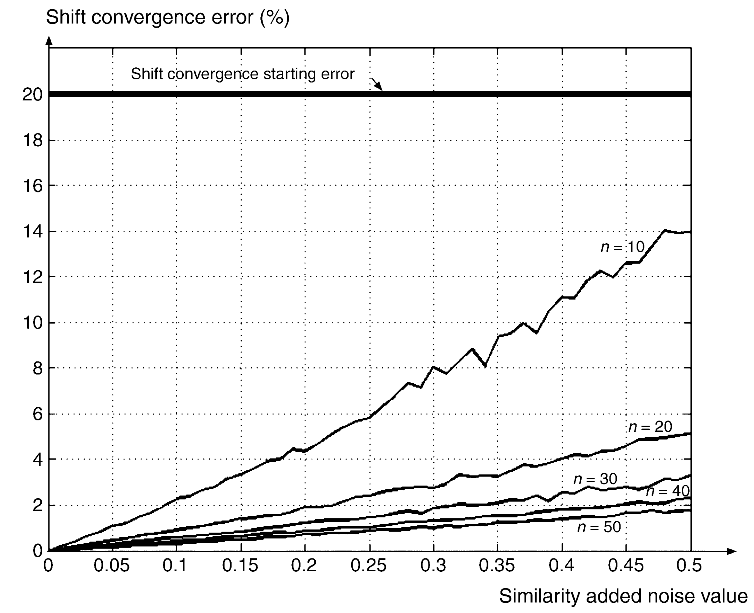


Figure 13 The influence of noise in shift calculation

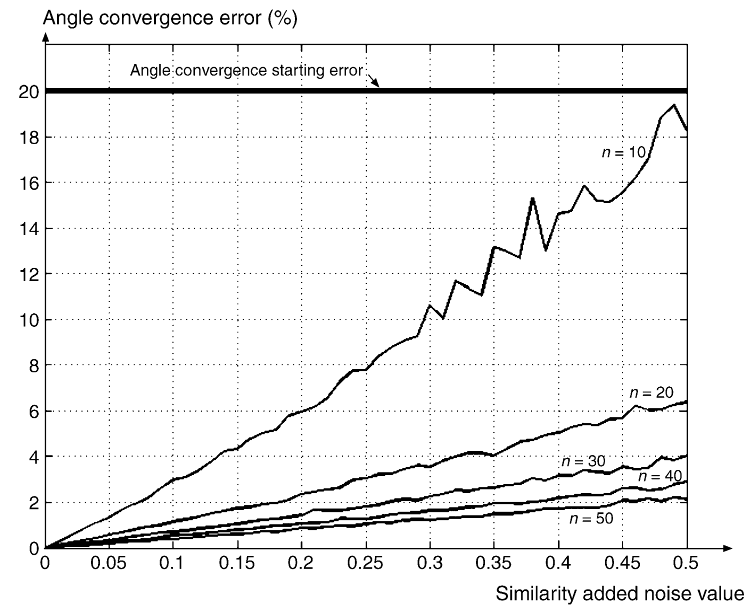


Figure 14 The influence of noise in angle calculation

6.3 Influence of Noise and Training Sample Size on Convergence Rate

We performed an additional test to examine the stability of our algorithm, and the results are visualized in Figure 15. The X axis represents the amplitude of added noise, while

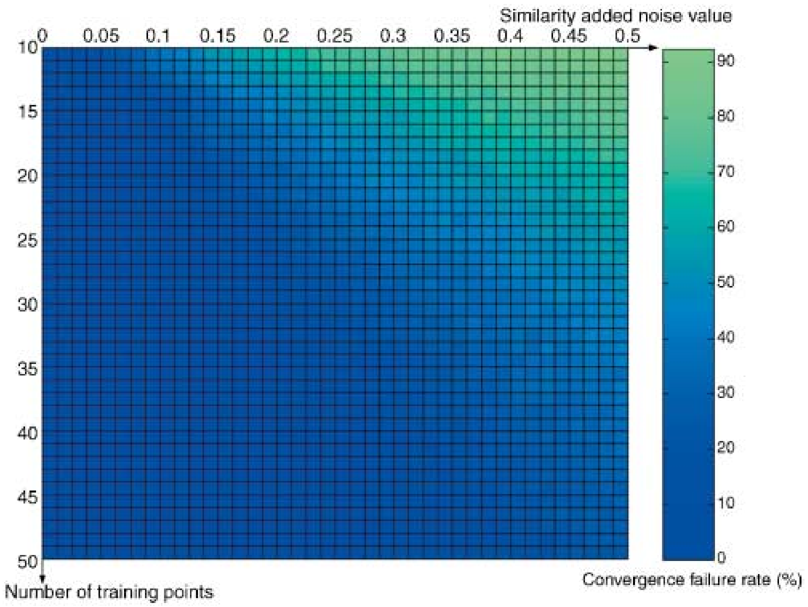


Figure 15 Influence of noise and training sample size for convergence rates
This figure appears in colour in the electronic version of this article and in the plate section at the back of the printed journal

the Y axis represents the total number of training points used. The reader should note that for this figure we have used a reverse color-coding scheme: light gray values correspond to high failure percentages in our convergence, while dark gray values indicate convergence success. We investigated the effect of noise and training size on convergence rate. The graph demonstrates a very robust behavior, with limited failure incidents. Convergence failure instances are mostly contained to the upper right part of the graph, corresponding to high noise and limited training points. In such instances convergence failure is attributable to the limited modeling capabilities of the sigmoidal function (e.g. it is a monotonically decreasing function). The variability within values is caused by the randomness of the selected point with which the noise is added.

6.4 Influence of Noise and Training Sample Size on Iteration Number

Even though the overall approach aims at improving retrieval accuracy the influence of noise and training sample size to the required iteration number was a concern. Therefore we performed the experiment below to assess how the average iteration number for successful convergence changes as noise increases. The results are presented in Figure 16. Noise is again inserted randomly to a single point of the training set as an error in the similarity value (X axis). We also included various training sizes (Y axis). We can see that the number of iterations increases when the number of training points decreases, as expected, but not to a prohibitively high number, since the difference is only a single additional iteration. The same conclusion applies to the introduction of noise, that of a single iteration difference. The above remarks show that our algorithm's training speed is not significantly affected by noise.

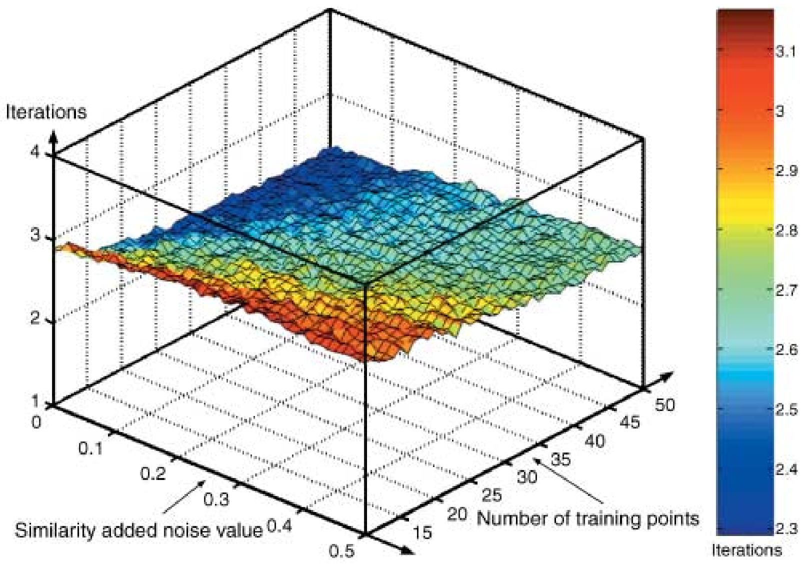


Figure 16 Influence of noise and training sample size for iteration number
This figure appears in colour in the electronic version of this article and in the plate section at the back of the printed journal

7 Conclusions

In this paper we investigated the application of user profiles as a means for more accurate geospatial information retrieval. We base our approach on our work on complex surface interpolation to model user preferences. Here, we expanded this early work to support the generation and use of user profiles for geospatial information retrieval. Of particular interest was the investigation of the statistical behavior of our system, considering three major factors, namely the accuracy of initial approximations, the effects of noise and the training sample size. The experiments presented in this paper confirmed that our system's behavior is consistent and can address noisy datasets. The overall convergence rate was excellent and the incorporation of multi-stage learning, where initial approximations take their values from previous less complex functions, proved its significance. These characteristics make our approach highly suitable for the increasingly diverse GIS user community, rendering it a valuable method for customized geospatial queries.

Acknowledgements

This work was supported by the Natural Science Foundation through grant ITR-0121269.

References

Aha D 1992 Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms. *International Journal of Man-Machine Studies* 36: 267–87

Atkeson C G, Moore A W, and Schaal S 1997 Locally weighted learning. *Artificial Intelligence Review* 11: 11–73

Batchelor B G 1978 *Pattern Recognition: Ideas in Practice*. New York, Plenum Press

Bennet D and Armstrong M 1996 An inductive based approach to terrain feature extraction. *Cartography and Geographic Information Systems* 23: 3–19

Biberman Y 1994 A context similarity measure. In *Proceedings of the European Conference on Machine Learning*, Catalina, Italy: 49–63

Carkacioglu A and Fatos Y V 2002 Learning similarity space. In *Proceedings of the International Conference on Image Processing*, Rochester, New York: 405–8

Cost S and Salzberg S 1993 A weighted nearest neighbor algorithm for learning with symbolic features. *Machine Learning* 10: 57–78

Diday E 1974 Recent progress in distance and similarity measures in pattern recognition. In *Proceedings of the Second International Joint Conference on Pattern Recognition*, Copenhagen, Denmark: 534–39

Domingos P 1995 Rule induction and instance-based learning: A unified approach. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, Montreal, Canada: 1226–32

Doucette P, Agouris P, Stefanidis A, and Musavi M 2001 Self-organized clustering for road extraction in classified imagery. *ISPRS Journal of Photogrammetry and Remote Sensing* 55: 347–58

Gionis A, Indyk P, and Motwani R 1999 Similarity search in high dimensions via hashing. In *Proceedings of the Twenty-fifth International Conference on Very Large Data Bases* (VLDB), Edinburgh, Scotland: 518–29

Ishii N and Wang Y 1998 Learning feature weights for similarity measures using genetic algorithms. In *Proceedings of the IEEE International Joint Symposium on Intelligence and Systems*, Rockville, Maryland: 27–33

Lance G N and Williams W T 1966 Computer programs for classification. In *Proceedings of the ANCCAC Conference*, Canberra, Australia (Paper 12/3)

Leydesdorff L 2004 Similarity measures, author cocitation analysis, and information theory. *Journal of the American Society for Information Science and Technology* 56: 769–72

Lim J H, Wu J K, Singh S, and Narasimhalu A D 2001 Learning similarity matching in multimedia content-based retrieval. *IEEE Transactions on Knowledge and Data Engineering* 13: 846–50

Ma W and Manjunath B S 1998 A texture thesaurus for browsing large aerial photographs. *Journal of the American Society of Information Science* 49: 633–48

Mandl T 2000 Tolerant information retrieval with backpropagation networks. *Neural Computing and Applications* 9: 280–9

Mitaim S and Kosko B 1998 Neural fuzzy agents for profile learning and adaptive object matching. *Presence* 7: 617–37

Mountrakis G and Agouris P 2003 Learning similarity with fuzzy functions of adaptable complexity. In Hadzilacos T, Manolopoulos Y, Roddick J, and Theodoridis Y (eds) *Advances in Spatial and Temporal Databases*. Berlin, Springer Lecture Notes in Computer Science No. 2750: 412–29

Nadler M and Smith E P 1993 *Pattern Recognition Engineering*. New York, John Wiley and Sons

Rachlin J, Kasif S, Salzberg S, and Aha D W 1994 Towards a better understanding of memory-based and Bayesian classifiers. In *Proceedings of the Eleventh International Machine Learning Conference*, New Brunswick, New Jersey: 242–50

Salzberg S 1991 A nearest hyperrectangle learning method. *Machine Learning* 6: 277–309

Santini S and Jain R 1999 Similarity measures. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 21: 871–83

Stanfill C and Waltz D 1986 Toward memory-based reasoning. *Communications of the ACM* 29: 1213–28

Tversky A 1977 Features of similarity. *Psychological Review* 84: 327–52

Vlachos M, Gunopulos D, and Kollios G 2002 Robust similarity measures for mobile object trajectories. In *Proceedings of the DEXA Workshops*, Aix-en-Provence, France: 721–8

Walker P and Moore D 1988 SIMPLE: An inductive modeling and mapping tool for spatially-oriented data. *International Journal of Geographical Information Systems* 2: 347–63

Wettschereck D, Aha D W, and Mohri T 1995 *A Review and Comparative Evaluation of Feature Weighting Methods for Lazy Learning Algorithms*. Washington, D.C., Naval Research

Laboratory, Navy Center for Applied Research in Artificial Intelligence Technical Report No AIC-95-012

Wilson D and Martinez T 1997 Improved heterogeneous distance functions. *Journal of Artificial Intelligence Research* 6: 1–34

Wilson D R and Martinez T R 2000 An integrated instance-based learning algorithm. *Computational Intelligence* 16: 1–28

Xu Y, Ruan D, and Liu J 1999 Uncertainty automated reasoning in intelligent learning of soft knowledge. In *Proceedings of the ICST Workshop Information and Communication Technology for Teaching and Training*, Gent, Belgium: 29–45

Yi B K and Faloutsos C 2000 Fast Time Sequence Indexing for Arbitrary Lp Norms. In *Proceedings of the Twenty-sixth International Conference on Very Large Data Bases* (VLDB), Cairo, Egypt: 385–94

Zadeh L A 1972 A fuzzy-set-theoretic interpretation of linguistic hedges. *Journal of Cybernetics* 2(3): 4–34